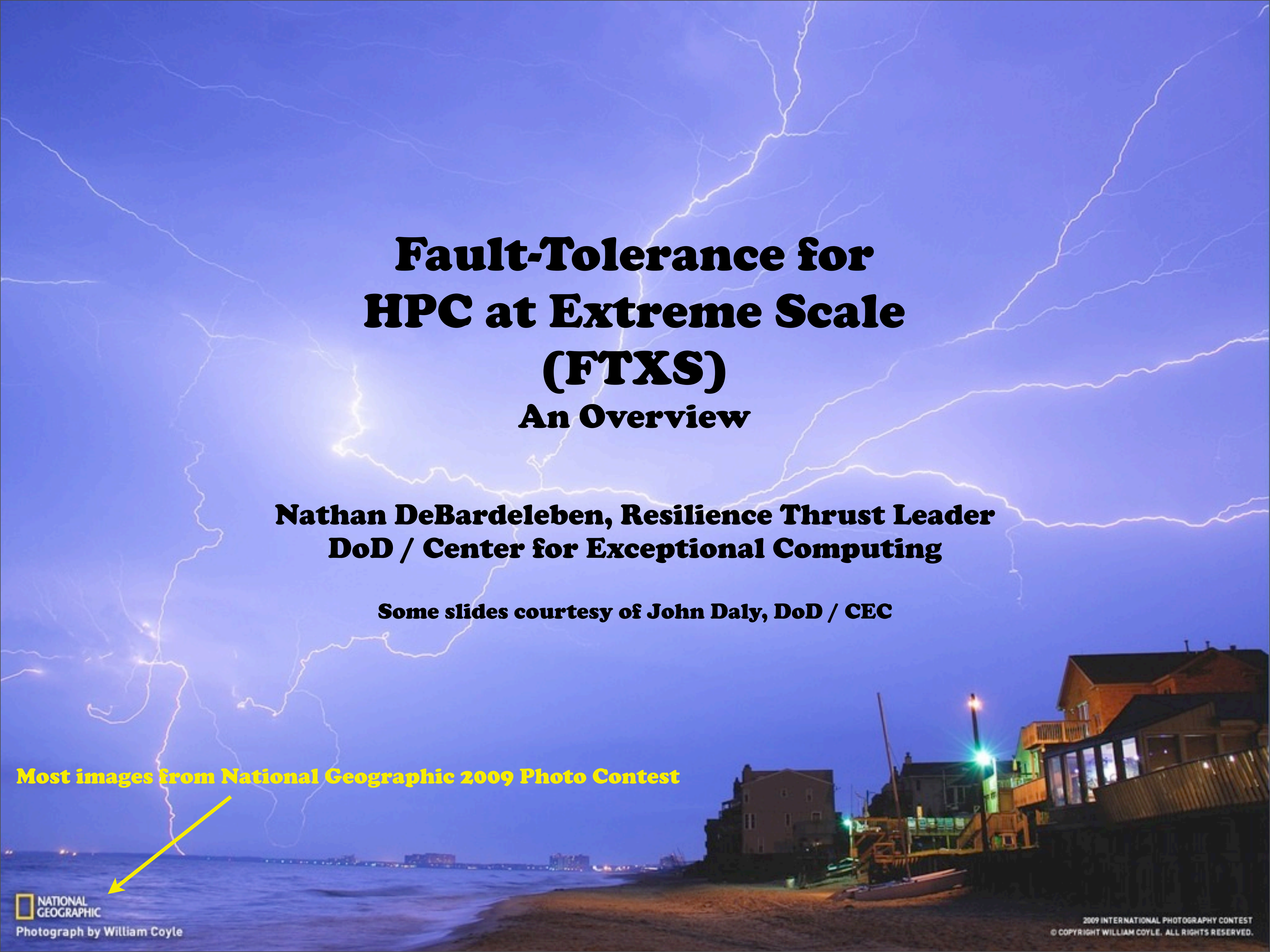

NATIONAL GEOGRAPHIC
Photograph by William Coyle
2009 INTERNATIONAL PHOTOGRAPHY CONTEST
© COPYRIGHT WILLIAM COYLE. ALL RIGHTS RESERVED.

Fault-Tolerance for HPC at Extreme Scale (FTXS)

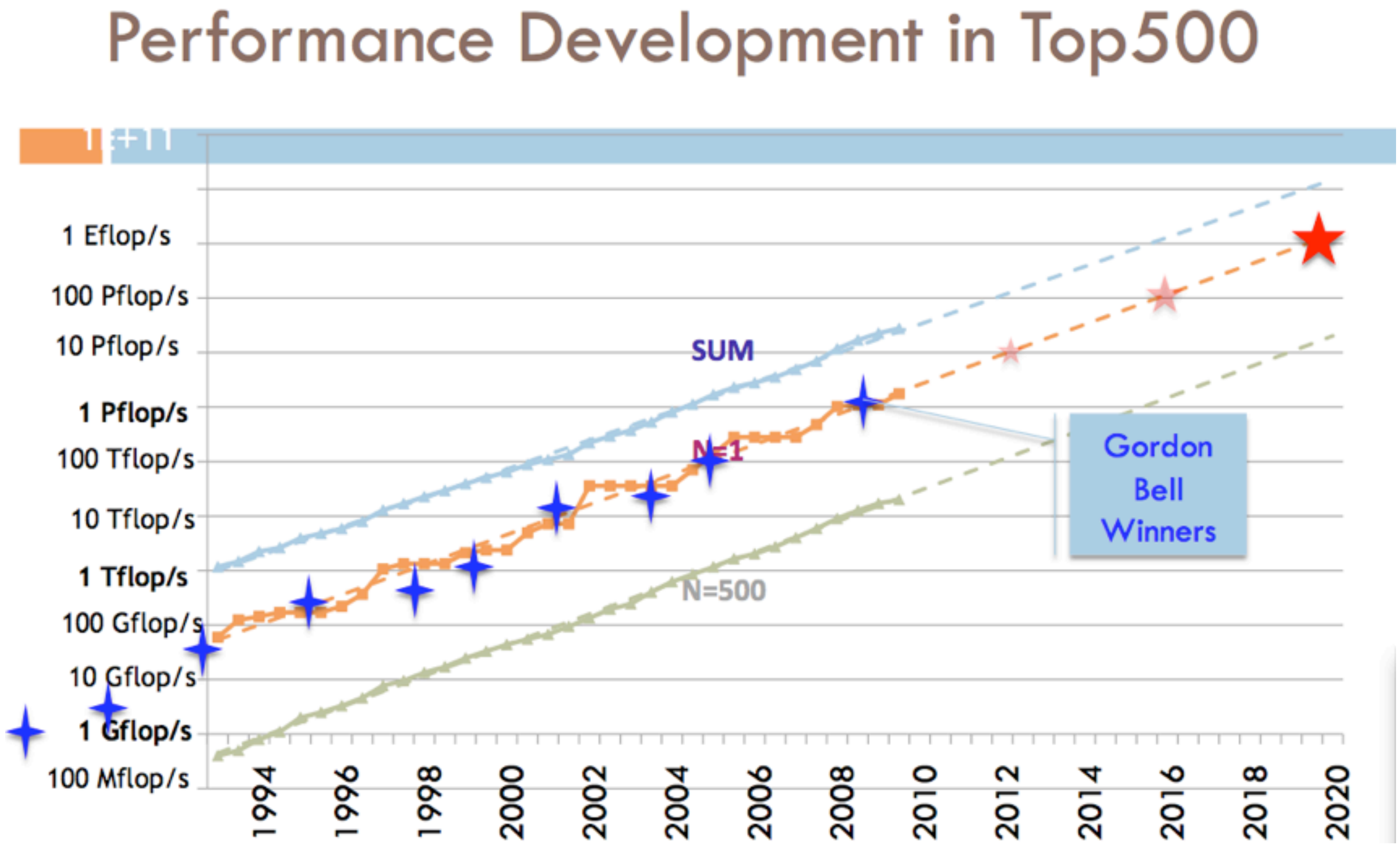
An Overview

Nathan DeBardeleben, Resilience Thrust Leader
DoD / Center for Exceptional Computing

Some slides courtesy of John Daly, DoD / CEC

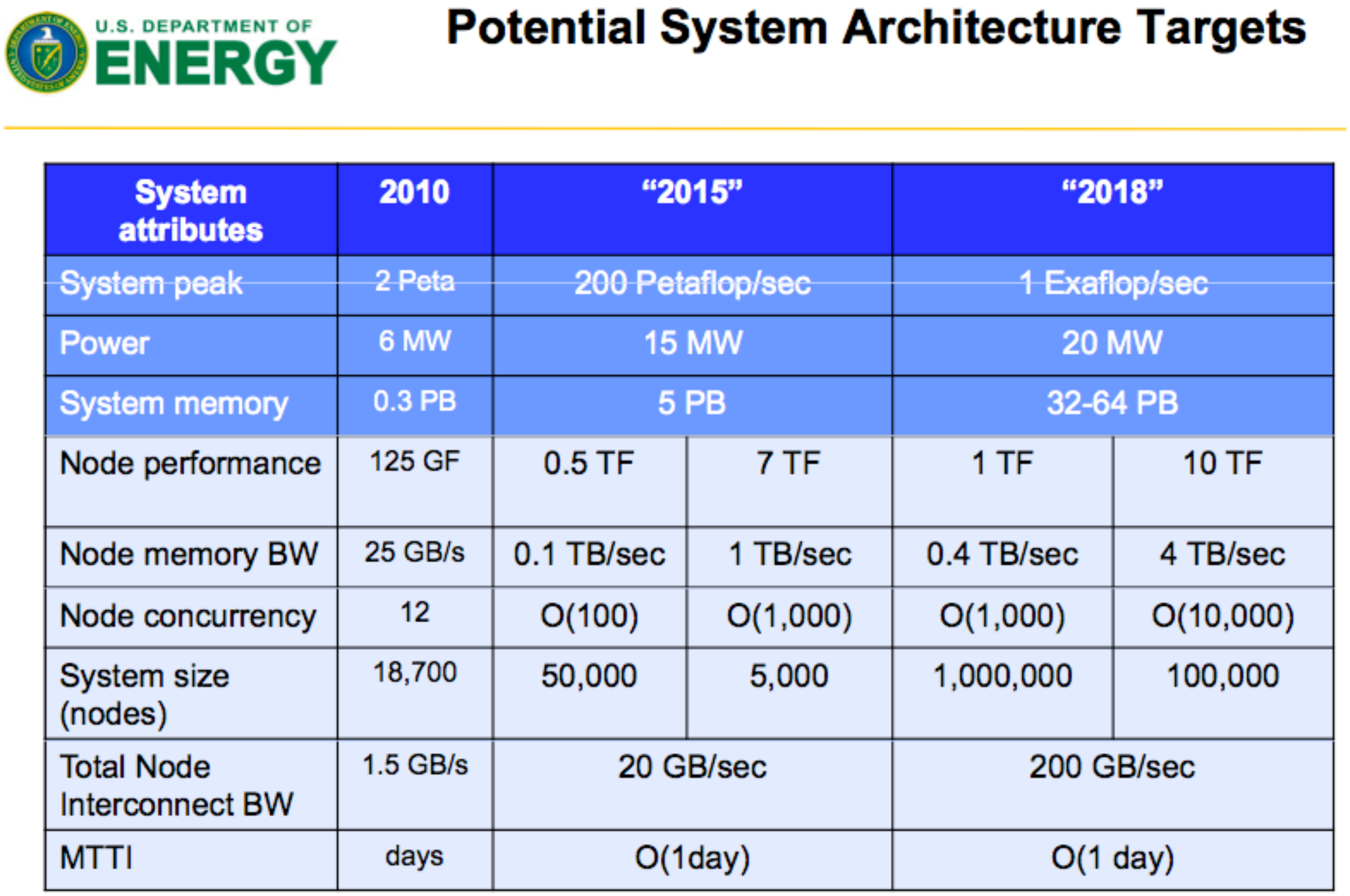
Most images from National Geographic 2009 Photo Contest

Where is HPC headed?
Projections based on the history of HPC performance



From Pete Beckman & Jack Dongarra, http://www.exascale.org

A typical DOE Exascale roadmap, but what should we make of the constant reliability numbers?

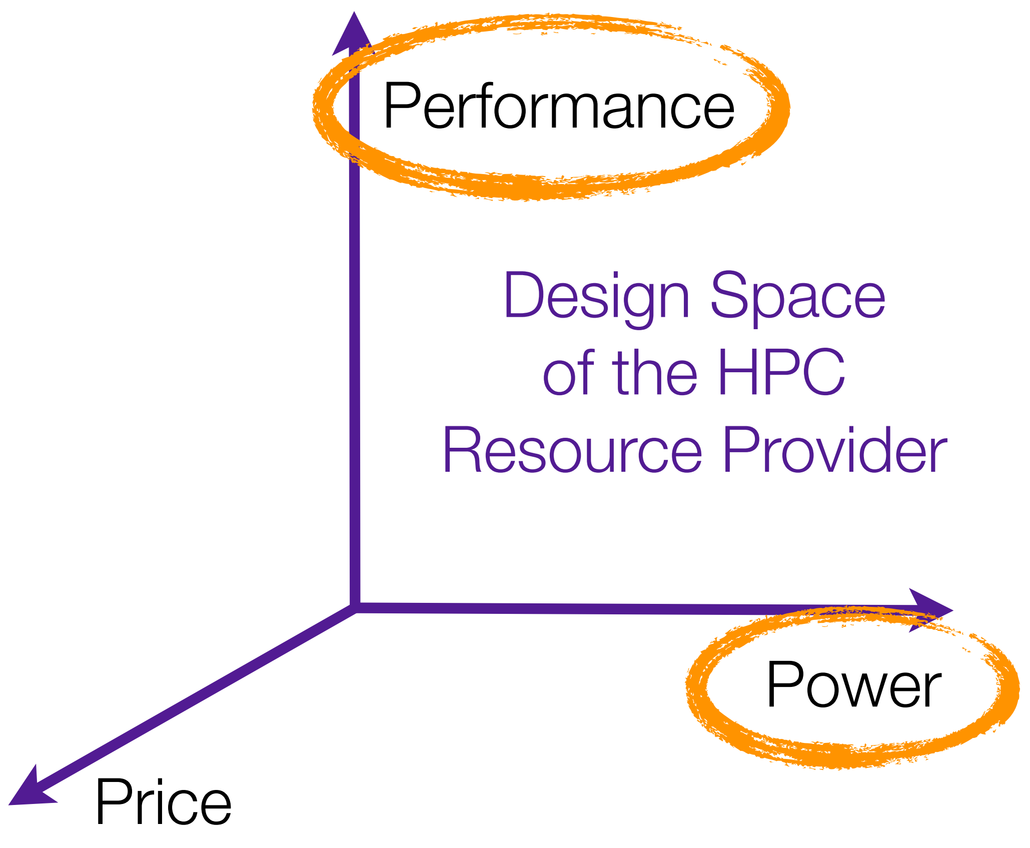


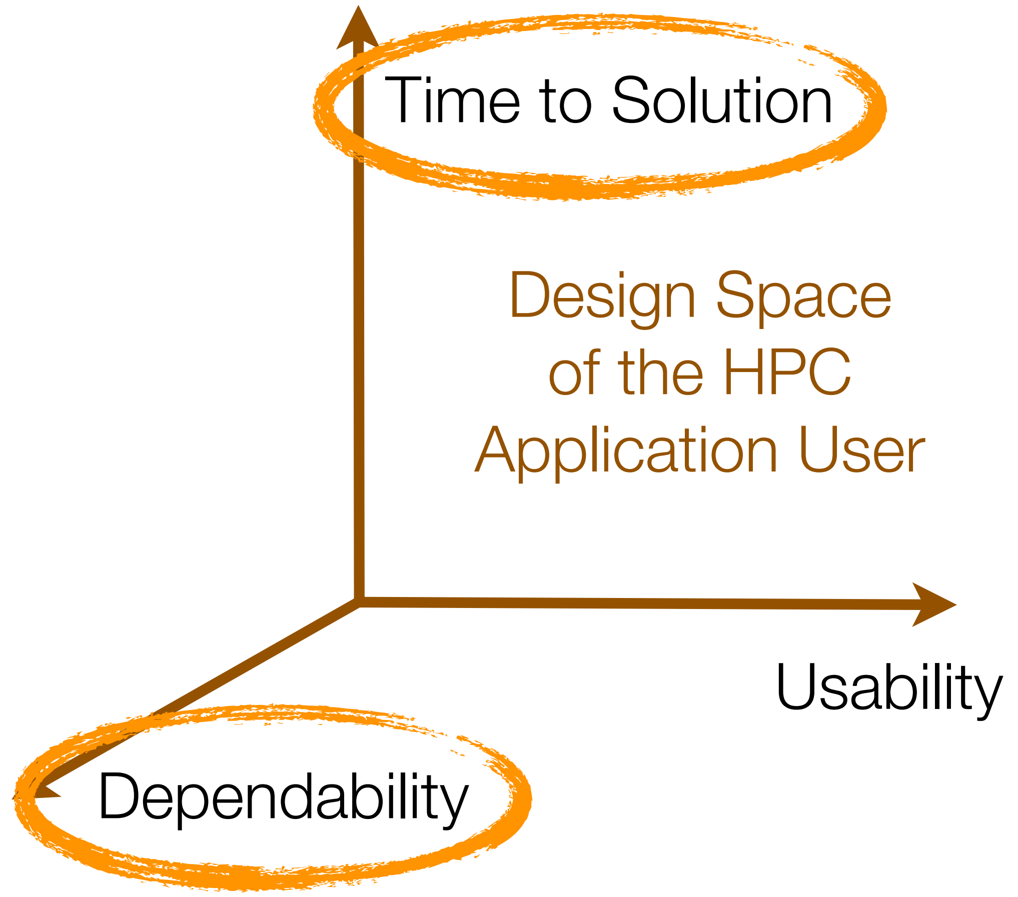
U.S. DEPARTMENT OF ENERGY

Potential System Architecture Targets

System attributes	2010	"2015"		"2018"	
System peak	2 Peta	200 Petaflop/sec		1 Exaflop/sec	
Power	6 MW	15 MW		20 MW	
System memory	0.3 PB	5 PB		32-64 PB	
Node performance	125 GF	0.5 TF	7 TF	1 TF	10 TF
Node memory BW	25 GB/s	0.1 TB/sec	1 TB/sec	0.4 TB/sec	4 TB/sec
Node concurrency	12	O(100)	O(1,000)	O(1,000)	O(10,000)
System size (nodes)	18,700	50,000	5,000	1,000,000	100,000
Total Node Interconnect BW	1.5 GB/s	20 GB/sec		200 GB/sec	
MTTI	days	O(1day)		O(1 day)	

Data movement is THE biggest challenge facing future HPC systems





- Performance challenge - 1 Exaflop @ 2 GHz / thread -> ~500 Mthreads
- Power challenge - 1 Exaflop @ 20 MWatts ->20 pJ/op to move data
- Reliability challenge - 1 day MTTI @ 500 Mthreads -> 10^9 hrs MTTI per thread

It is more than just the increase in the number of components driving up the fault rate

- **Number of components** both memory and processors will increase by an order of magnitude which will increase hard and soft errors
- **Smaller circuit sizes, running at lower voltages** to reduce power consumption, increases the probability of switches flipping spontaneously due to thermal and voltage variations as well as radiation, increasing soft errors
- **Power management cycling** significantly decreases the components lifetimes due to thermal and mechanical stresses.
- **Resistance to add additional HW detection and recovery logic** right on the chips to detect silent errors. Because it will increase power consumption by 15% and increase the chip costs.
- **Heterogeneous systems** make error detection and recovery even harder, for example, detecting and recovering from an error in a GPU can involve hundreds of threads simultaneously on the GPU and hundreds of cycles in drain pipelines to begin recovery.
- **Increasing system and algorithm complexity** makes improper interaction of separately designed and implemented components more likely.
- **Number of operations** (10^{23} in a week) ensure that system will traverse the tails of the operational probability distributions.

NATIONAL GEOGRAPHIC
Photograph by Gary Norbraten
2009 INTERNATIONAL PHOTOGRAPHY CONTEST
© COPYRIGHT GARY NORBRATEN. ALL RIGHTS RESERVED.

Thanks to Al Geist, ORNL and Sudip Dosanjh, SNL

What is resilience?

Resilience is an approach to dealing with failures that focuses on keeping the applications running

NATIONAL GEOGRAPHIC
Photograph by Don Love
2009 INTERNATIONAL PHOTOGRAPHY CONTEST
© COPYRIGHT DON LOVE. ALL RIGHTS RESERVED

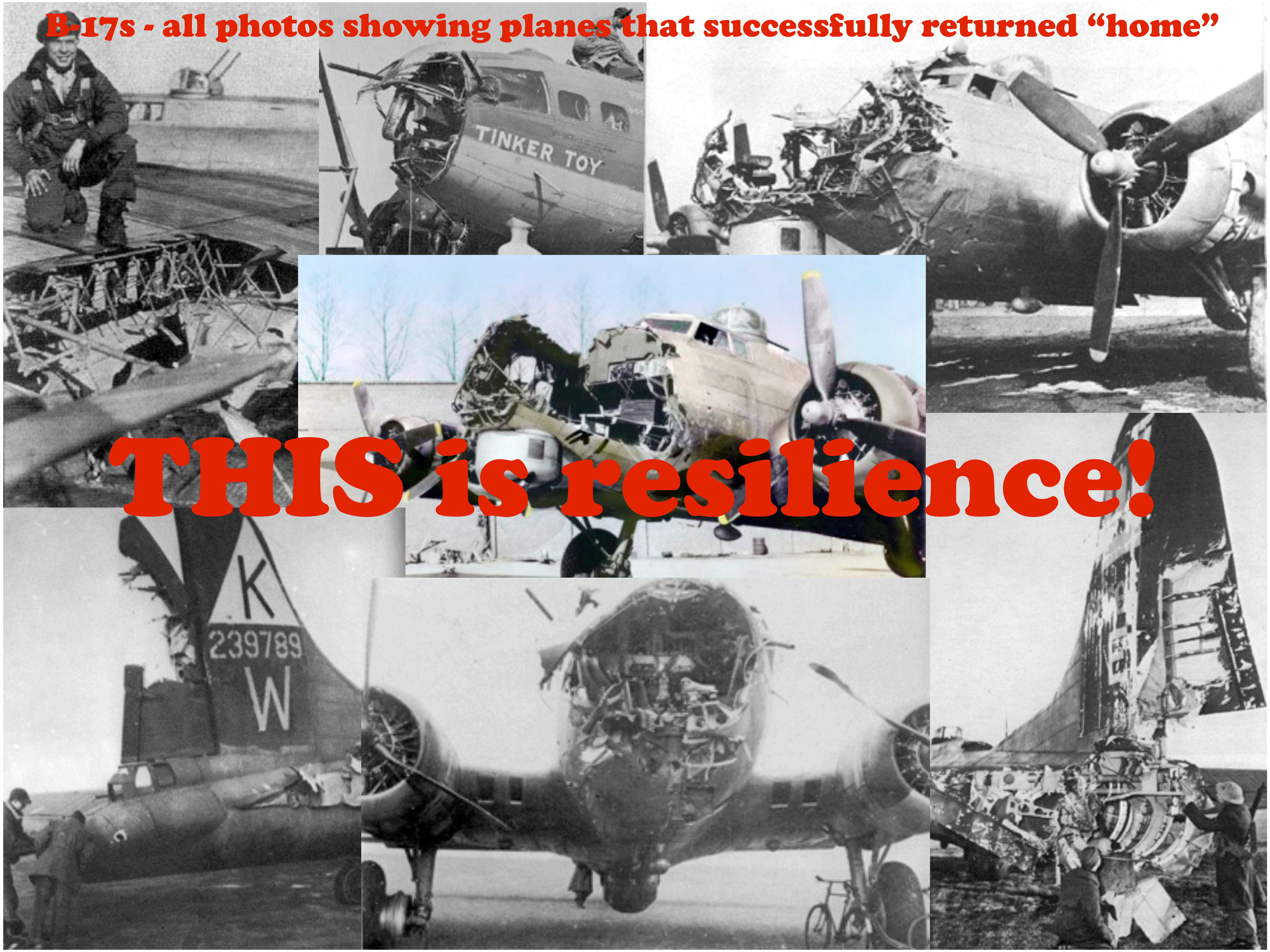
- Resilience is concerned with **platforms** and **applications** of extreme scale or complexity where failures are frequent and highly interdependent
- Resilience is dynamic and adaptive, leveraging tools from fault-tolerance with **platform** and **application** feedback to respond to and even preempt failures
- Resilience recognizes that the state of the **application** may not be easily or uniquely determined by the state of the **platform**
- Resilience "embraces failure" by accepting that **platform** errors will occur and focusing resources on reducing or eliminating **application** errors

AMTTI >> SMTBF

Fault-tolerance vs. Resilience

- Resilience is dynamic and adaptable to an evolving system environment
- Resilience tries to keep the application out of trouble, but also tries to help when an application gets into trouble
- Resilience is a quality of service agreement between a platform provider and an application user: *"I cannot promise that nothing bad will happen if you run on our systems, but I will provide necessary and reasonable safeguards!"*

B-17s - all photos showing planes that successfully returned "home"



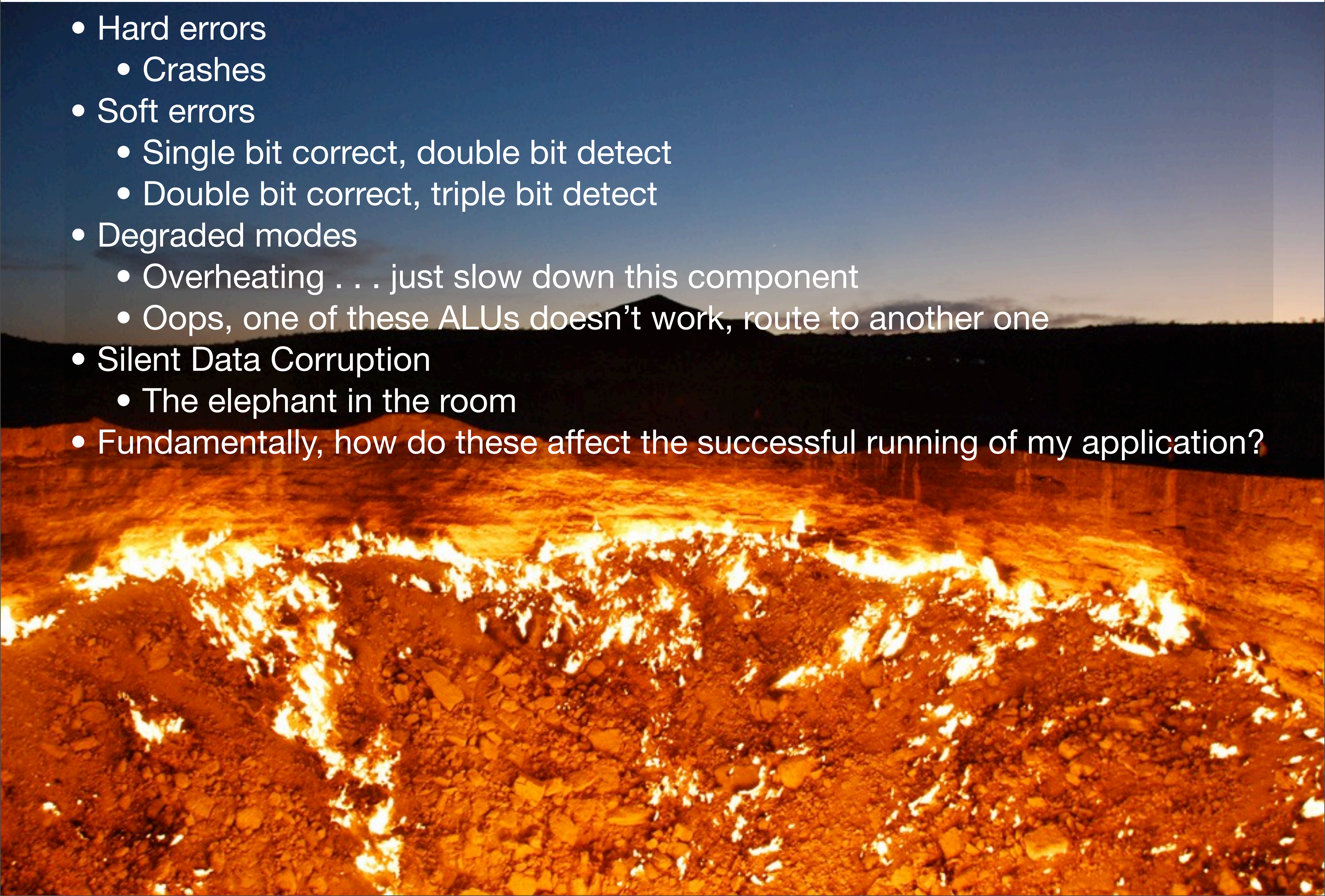
THIS is resilience!

. . . but so is this

EMBRACE FAILURE

AVOID FAILURE

HPC Reliability - What's really going on?



- Hard errors
  - Crashes
- Soft errors
  - Single bit correct, double bit detect
  - Double bit correct, triple bit detect
- Degraded modes
  - Overheating . . . just slow down this component
  - Oops, one of these ALUs doesn't work, route to another one
- Silent Data Corruption
  - The elephant in the room
- Fundamentally, how do these affect the successful running of my application?

What really matters: Application-centric View

- What does an application designer / programmer do to mitigate some of these failure modes?
- To what extent is my application resistant to:
  - Communication corruption?
  - Temporary disconnection?
  - Transient hardware failures?
  - Damaged software?
- The application developer today does not have all the necessary tools to answer these questions
- What tools would be required?

What is industry best practice?

- Save application state . . . somewhere . . . to try and reduce the affect of failure so that we can recover
- “Checkpointing” and “Recovery”
  - Global shared state to persistent storage
  - In-memory checkpointing gaining traction
  - Buffering with SSDs / NVRAM
- Application entirely take on this burden themselves
  - Hand-coded checkpointing algorithms
  - Some libraries exist (notably BLCR) . . . but portable?
  - Optimal checkpoint interval?
    - For this machine . . . and this code . . . with this current failure characteristic . . .
- Any way you cut it, it is defensive I/O
  - And I/O systems can fail too

Exascale will require new levels of reliability across every level of the system

U.S. DEPARTMENT OF ENERGY

Need solutions for decreased reliability and a new model for resiliency

- **Barriers**
  - System components, complexity increasing
  - Silent error rates increasing
  - Reduced job progress due to fault recovery if we use existing checkpoint/restart
- **Technical Focus Areas**
  - Local recovery and migration
  - Development of a standard fault model and better understanding of types/rates of faults
  - Improved hardware and software reliability
    - Greater integration across entire stack
  - Fault resilient algorithms and applications
- **Technical Gap**
  - Maintaining today's MTTI given 10x - 100X increase in sockets will require:

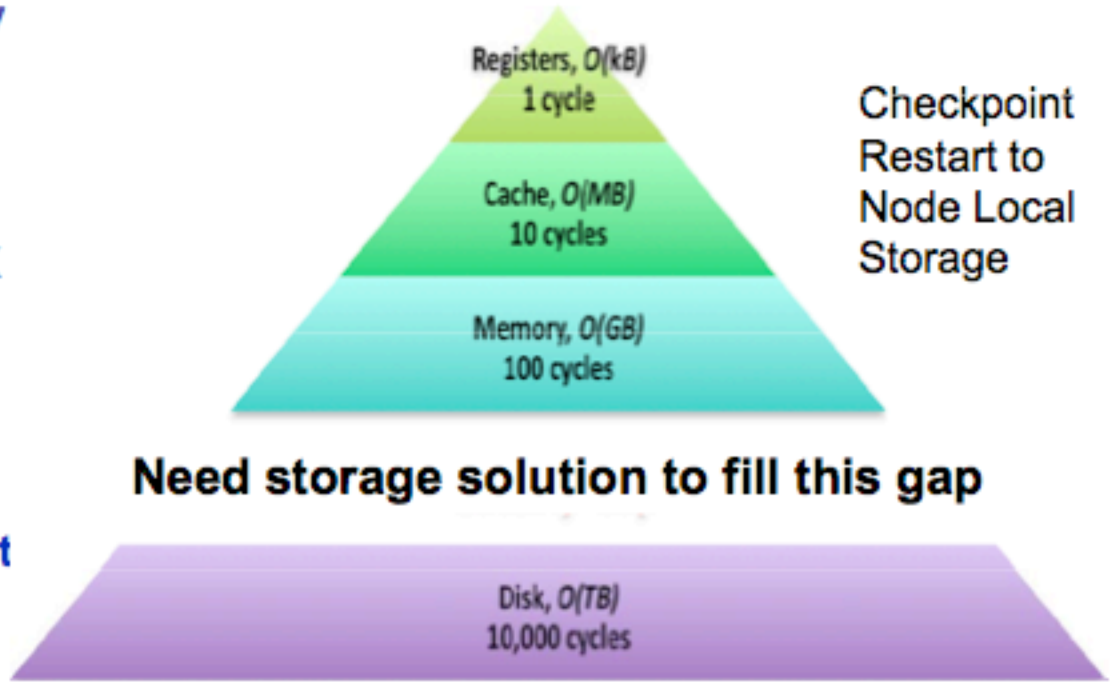
    10X improvement in hardware reliability

    10X in system software reliability, and

    10X improvement due to local recovery and migration as well as research in fault resilient applications
  - .

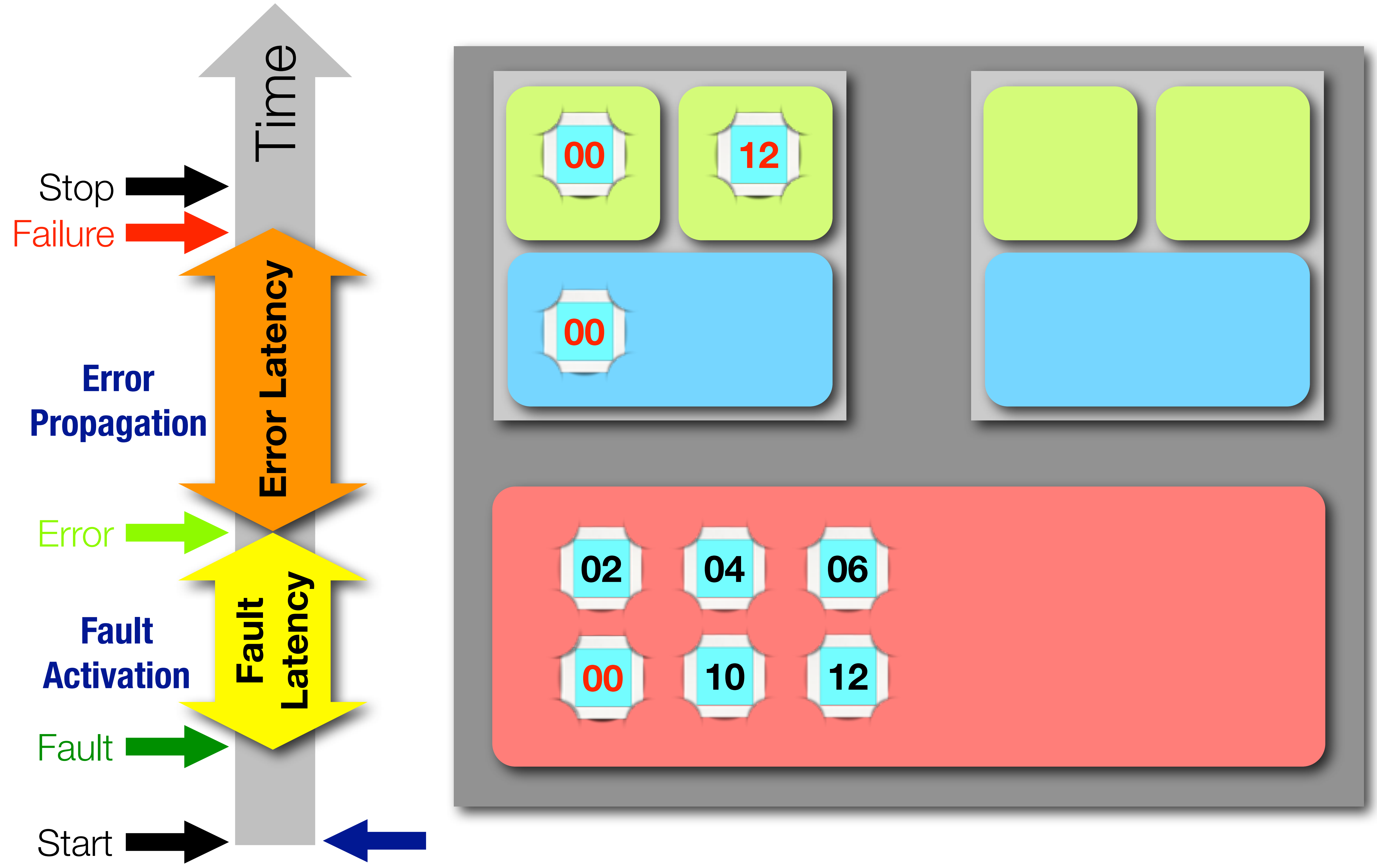
Taxonomy of errors (h/w or s/w)

- **Hard errors**: permanent errors which cause system to hang or crash
- **Soft errors**: transient errors, either correctable or short term failure
- **Silent errors**: undetected errors either permanent or transient. *Concern is that simulation data or calculation have been corrupted and no error reported.*

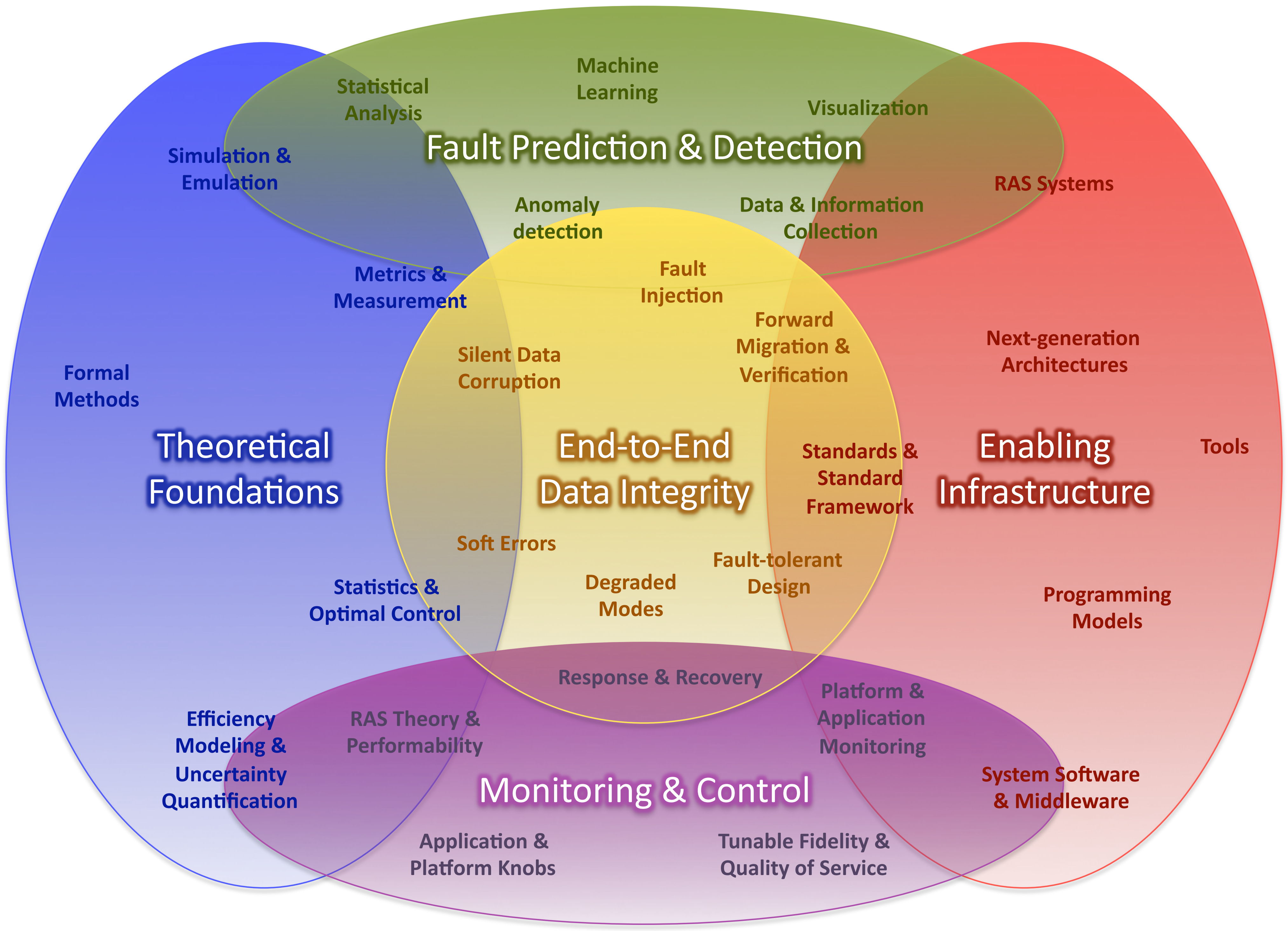


Exascale Initiative Steering Committee

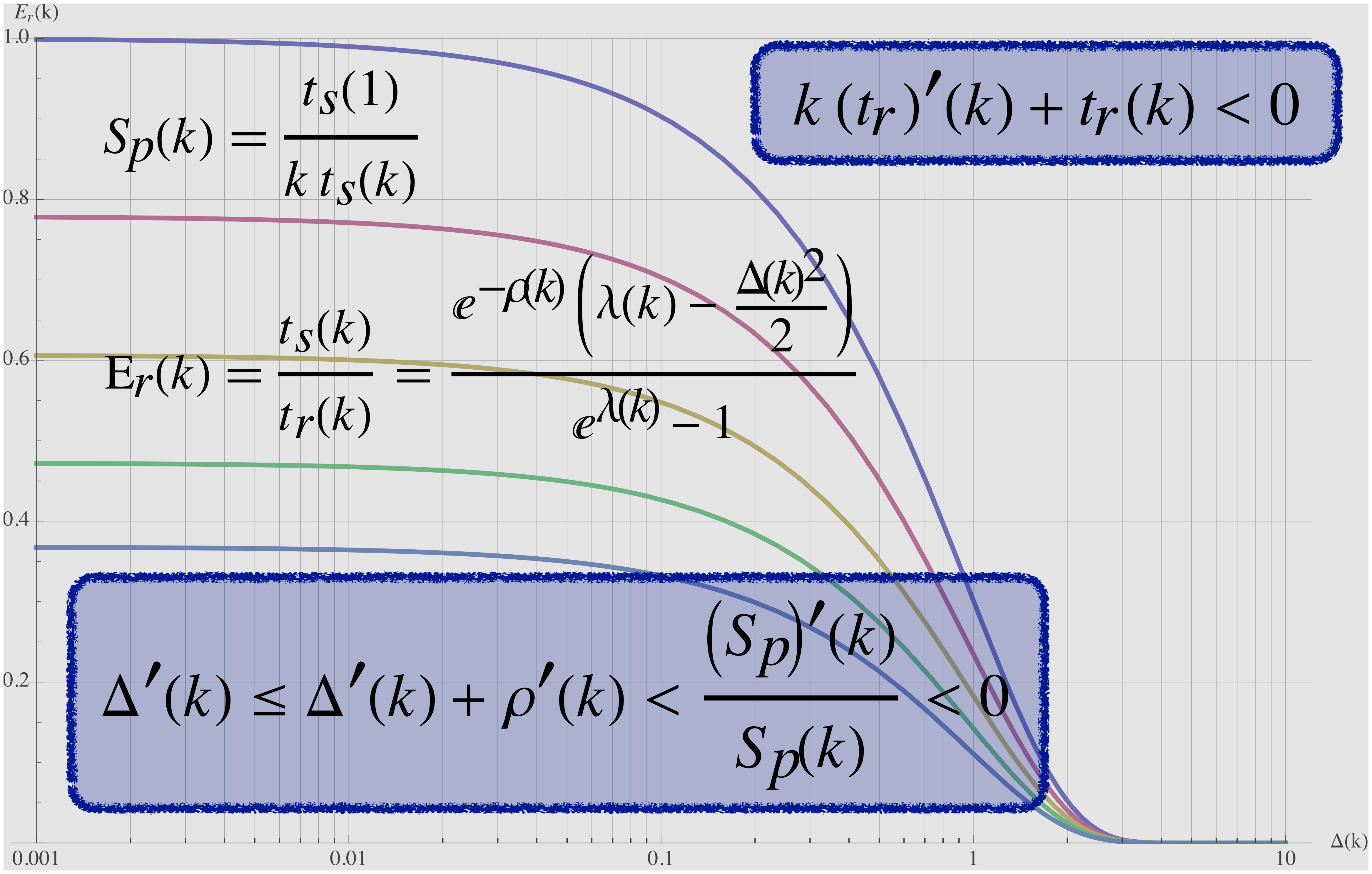
Why is this so hard? Illustrating the timeline of fault, error and failure in the execution of an application



Resilience has a massive scope: five overlapping thrusts with multiple key areas for focused R&D


Machine Learning
Statistical Analysis
Visualization
Fault Prediction & Detection
Simulation & Emulation
RAS Systems
Anomaly detection
Data & Information Collection
Fault Injection
Metrics & Measurement
Forward Migration & Verification
Next-generation Architectures
Silent Data Corruption
Formal Methods
Theoretical Foundations
End-to-End Data Integrity
Standards & Standard Framework
Enabling Infrastructure
Tools
Soft Errors
Fault-tolerant Design
Statistics & Optimal Control
Degraded Modes
Programming Models
Response & Recovery
Platform & Application Monitoring
Efficiency Modeling & Uncertainty Quantification
RAS Theory & Performability
System Software & Middleware
Monitoring & Control
Application & Platform Knobs
Tunable Fidelity & Quality of Service

Theoretical foundations: model failure and quantify "resource-efficient" computing



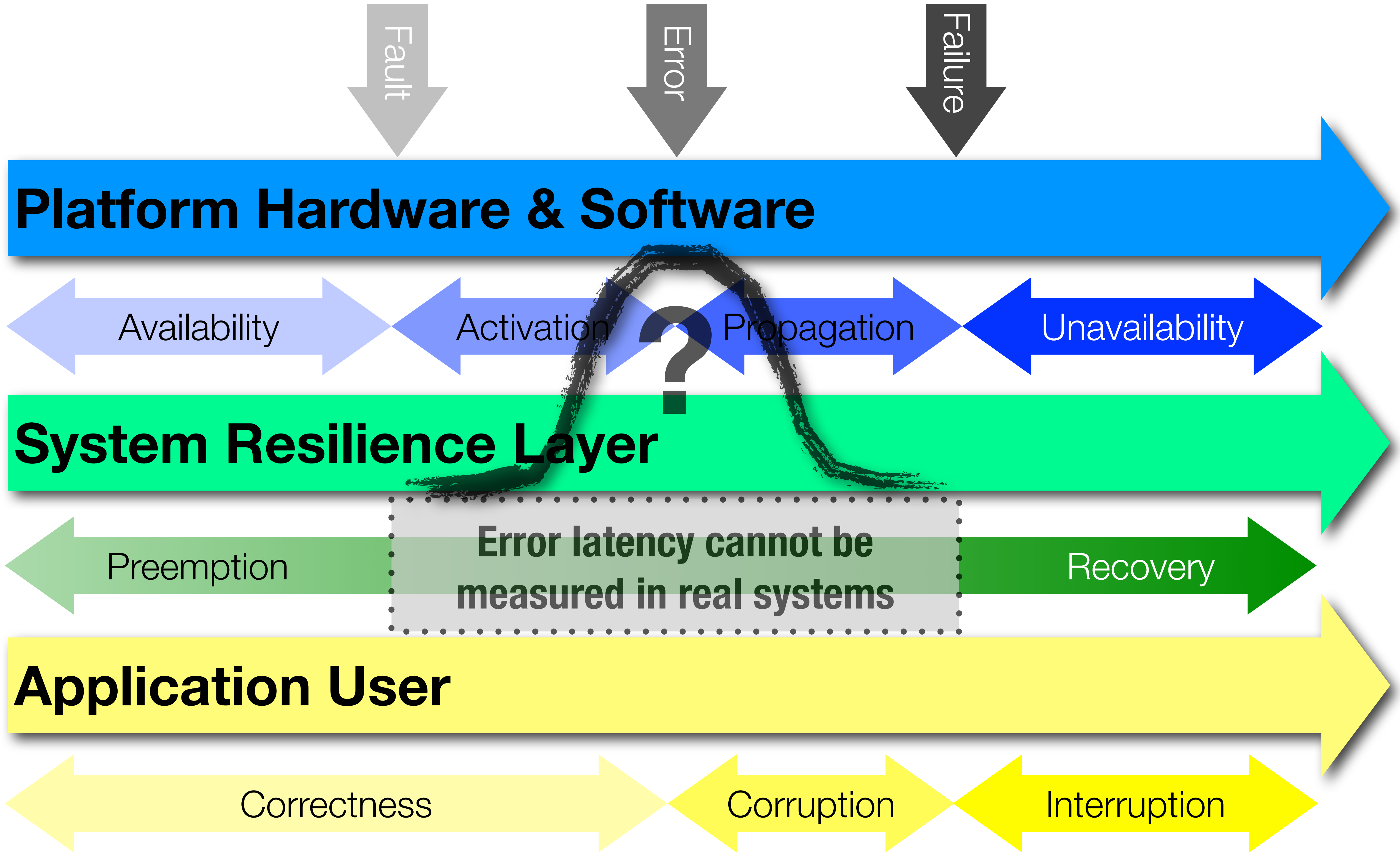
$$S_p(k) = \frac{t_s(1)}{k\, t_s(k)}$$

$$k\,(t_r)'(k) + t_r(k) < 0$$

$$\mathrm{E}_r(k) = \frac{t_s(k)}{t_r(k)} = \frac{e^{-\rho(k)}\left(\lambda(k) - \frac{\Delta(k)^2}{2}\right)}{e^{\lambda(k)} - 1}$$

$$\Delta'(k) \le \Delta'(k) + \rho'(k) < \frac{(S_p)'(k)}{S_p(k)} < 0$$

Enabling Infrastructure: getting harder as the gap between faults and failures grows

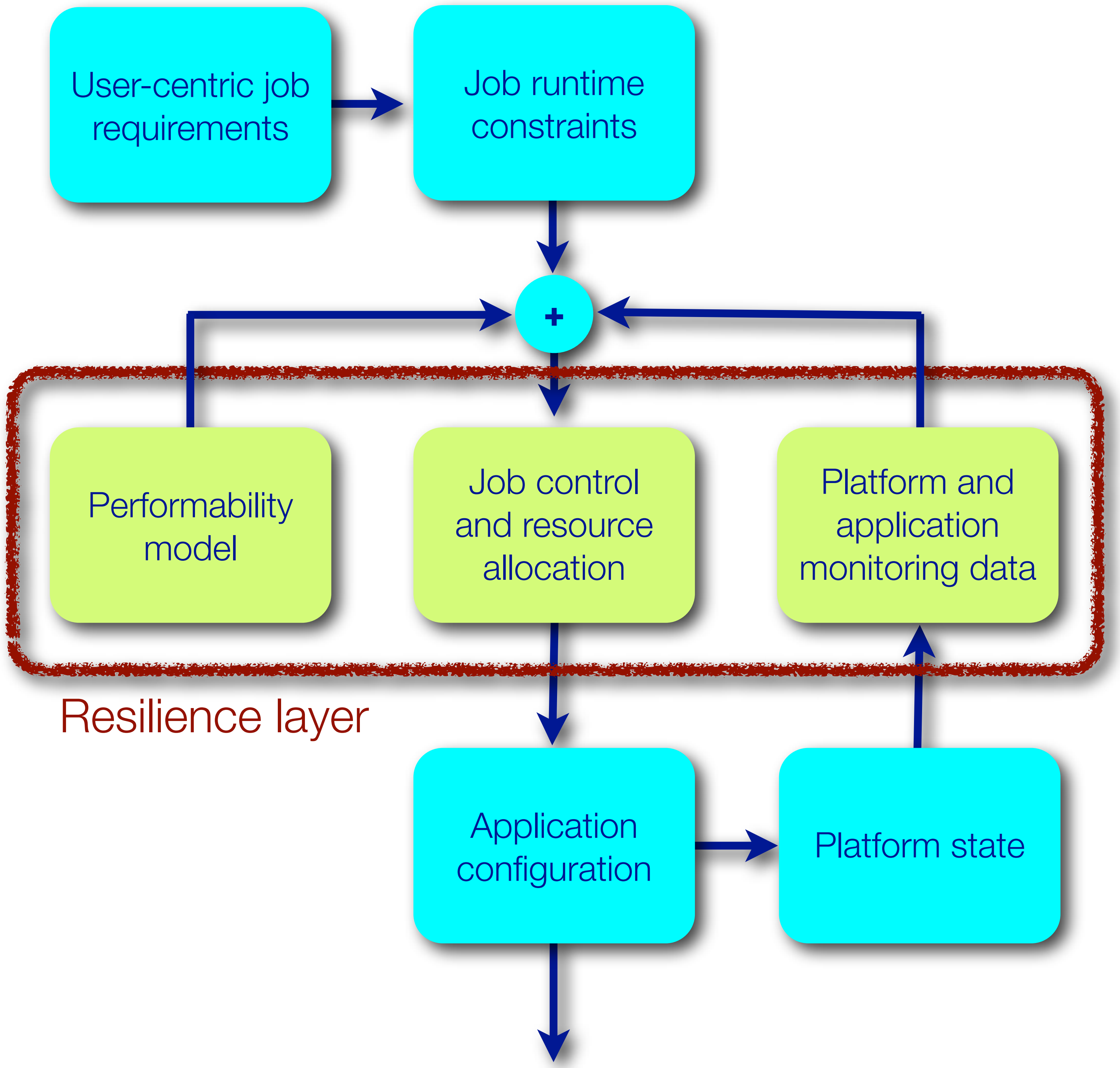


Fault Prediction & Detection: error latency only measurable by modeling and simulation

- Three questions to be answered with regards to the error latency:
  - **Scope**: Where is the error manifest?
  - **Duration**: How long did the error lie undetected?
  - **Extent**: How far did the error propagate?
- Redundancy cannot be executed confidently without a method of quantifying the spatial and temporal **scope**, **duration**, and **extent** of an error
- Result of modeling and simulation is the likelihood that a particular spatial or temporal domain of the system is uncorrupted
- **Tunable Fidelity**: Some systems may be able to sacrifice correctness in a particular domain if they can exchange if for power or performance

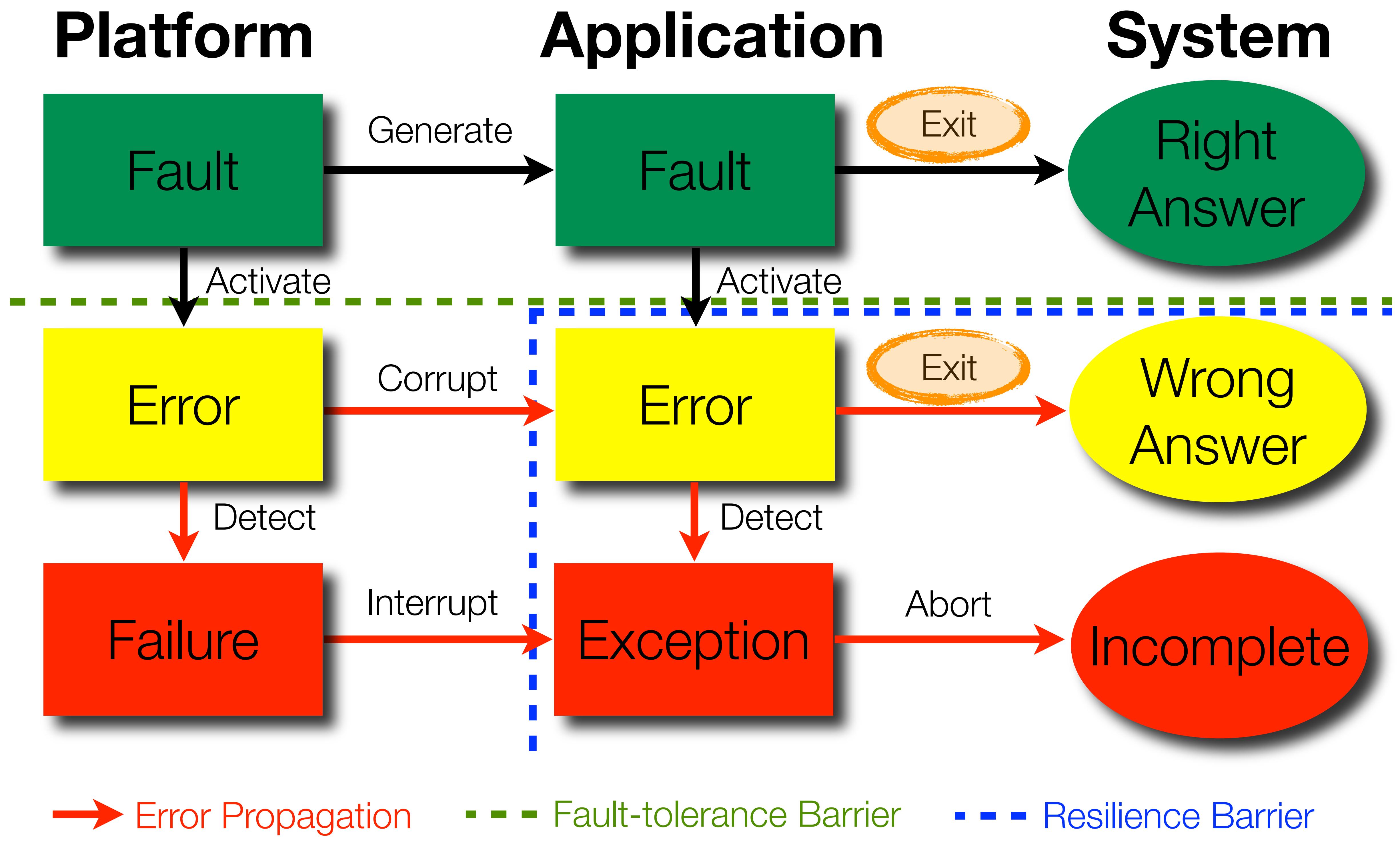
Monitoring & Control: requires the application and platform interaction through standard interfaces

- Detection plays an important role in a feedback and control approach to resilience
- Both platform and application data are required for failure detection (and probably for fault prediction also)
- A resilience layer provides a common interface between the platform and application in a resilient system
- Undetectable errors handled probabilistically by simulation and modeling



User-centric job requirements
Job runtime constraints
+
Performability model
Job control and resource allocation
Platform and application monitoring data
Resilience layer
Application configuration
Platform state

End-to-End Data Integrity: compared to traditional fault-tolerance, resilience is a new approach



SELSE
Silicon Errors in Logic - System Effects

- 40 year peak in neutron flux (10% higher than the long-term average)
- Cosmic rays offered as Toyota acceleration / braking cause . . . not substantiated
- “The companies most interested in soft errors are those who have lost money because of it”
  - Reactive, not proactive
- Soft error rates of latches approaching that of SRAM
- Devices are being placed too closely by the routing algorithms. Signals are “bleeding” into each other causing “noise” and latches to flip
- Moving to lead-free solder is helping with alpha particle scattering (this is debated)
- Aging, gate oxide degradation, and metal migration
- Soft error rates for SRAM back on the rise as we shrink from 60nm to 45nm
- For each 10 millivolt decrease in power supply voltage expect a 30% increase in soft error rate
- Multi-bit flips are becoming more prevalent
- Soft error community targeting a 10% performance overhead for dealing with soft errors - time to integrate those into your performance predictions!
- “People are leveraging power for reliability and they are really in for a shock”

NATIONAL GEOGRAPHIC
Photograph by Robert Garrett
2009 INTERNATIONAL PHOTOGRAPHY CONTEST
© COPYRIGHT ROBERT GARRETT. ALL RIGHTS RESERVED.

Many open research questions . . .

- Accurate fault prediction
- Improved error detection
- Established mapping between **platform** and **application errors**
    - What **IS** the state of an application?
    - How can we tell if an application “succeeds”?
- Reduction of recovery and migration latencies
- Preemption instead of recovery
- Standards, metrics and data
- What is the right amount of data to log? What data?
- Where can we use replication cost-effectively?
- Resilience benchmark for new systems . . . and existing systems
- How to decide which software component handles a detected problem?
- Can tightly coupled numerical simulations be converted to transactional model?
- Where can we use Hadoop?
- Is checkpointing “dead”?
- In what ways can we take advantage of non-volatile memory?
- . . . just to name a few!

Welcome to FTXS 2010! Questions?

Goat
NATIONAL GEOGRAPHIC
Photograph by Nigel Hilliert
2009 INTERNATIONAL PHOTOGRAPHY CONTEST
© COPYRIGHT NIGEL HILLIER. ALL RIGHTS RESERVED.